|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | Кафедра математического и программного обеспечения ЭВМ |

# РАЗРАБОТКА АБСТРАКТНЫХ ТИПОВ ДАННЫХ

|  |  |
| --- | --- |
| **Дисциплина:** | ООП |
| **Темы:** | Инкапсуляция; классы; поля и компонентные функции; конструкторы и деструкторы класса |

**Среда разработки:** Microsoft Visual Studio

**Язык программирования:** C++

**Тип проекта:** Консольное приложение

**Задание на лабораторную работу №1**

**ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

1. Каждый класс должен быть оформлен в отдельных файлах: заголовочный (.h) и файл с кодом (.cpp).
2. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
3. Придерживайтесь принципа DRY (Don’t repeat yourself).
4. Обязательно наличие комментариев.

**ЦЕЛЬ РАБОТЫ**

Изучить способы определения классов, правила доступа к элементам; приобрести практические навыки работы с объектами класса; изучить принципы и механизмы создания абстрактных типов данных.

Изучить синтаксис и семантику определения и вызова конструкторов и деструкторов. Приобрести практические навыки создания и уничтожения объектов. Изучить особенности применения различных видов конструкторов.

**ЗАДАНИЯ**

*8 часов*

1. Ознакомьтесь с УМП по ООП часть 1 раздел IV.
2. Разработайте алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (см. прил. 1):
   1. Предусмотрите закрытую реализацию и открытый интерфейс.
   2. Интерфейс должен содержать функции вывода содержимого и функции доступа к некоторым полям.
   3. Элементы данных должны быть представлены из 5 полей, один из которых должен быть статическим, второй иметь тип указатель, а третий тип перечисление, остальные на свое усмотрение.
   4. Примените основные виды конструкторов и определитесь с деструктором класса.
   5. Проверьте работоспособность АТД на тестовом наборе данных.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Дайте определение понятия «класс». Сформулируйте правила доступа к его элементам.
2. С какой целью в классе объединены компонентные данные и компонентные функции?
3. Каким образом осуществляется доступ к открытым и закрытым элементам?
4. Опишите назначение дружественных функций, назовите их разновидности.
5. Что понимается под указателем this?
6. Каковы особенности использования статических компонентных данных?
7. В чем заключается синтаксис и семантика компонентных функций static и const?
8. Каким образом могут изменяться компонентные данные объектов, объявленных константами?
9. Каковы особенности создания вложенных классов?

**ПРИМЕЧАНИЯ**

* Придерживайтесь следующей конвенции кода:
  + имена пользовательских типов с заглавной буквы;
  + имена компонентных данных класса и функций со строчной буквы;
  + имена компонентных данных класса можно в конце помечать символом «\_»;
  + не использовать транслитерацию, все названия должны быть на английском языке;
  + не использовать простые имена в названиях членов классов;
  + помечать компонентные функции ключевым словом *const,* если они гарантируют иммутабельность значений компонентных данных класса;
  + объявлять константными ссылочные параметры компонентных функций, для обеспечения неизменяемости данных, на которые они ссылаются, аналогично с указателями.
* Желательно использовать в АТД компонентные данные с типом указатель на символ (char\*) – для задания строки;
  + Следует помнить, что типы указатель, условно состоят из двух адресов памяти: свой и тот, на который они указывают.
  + Передача параметра типа указатель в функцию, ведет себя как передача параметра по значению, но с существенный отличием:
    - если передавать параметр по значению в функцию, в памяти создается его полная копия (все-равно что локальная переменная), меняя его значение мы меняем лишь значение его копии, оставляя нетронутым оригинал;
    - аналогично происходит и с указателем, в памяти так же создастся его копия (новый указатель с новым своим адресом в памяти), но адрес, на который он указывает остается тем же, каким он был у оригинального указателя.

Такое поведение является небезопасным, передавая указатель параметром в функцию мы можем изменить данные, на которые он указывает. Поэтому для обеспечения сохранности данных (если так требуется) используется константные указатели. Наглядный тому пример: литерал ***“Это строка”*** - имеет тип не ***char\*,***а***const char\****т.к. очевидно, что литералы неизменяемые.

* + Важно, во избежание проблем с компонентными данными класса типа ***char\*****,* проводите их полную инициализацию внутри компонентной функции класса, а не используйте простое присваиваниям им адреса входного параметра типа ***char\*.*** Это гарантирует безопасность объектов самого класса, в частности передачи их в качестве параметра компонентной функции по значению.
  + Во избежание утечки памяти при использовании полной инициализации указателя (ключевое слово ***new***) не забывайте, что такое выделение памяти требует и его и ручное освобождение (ключевое слово ***delete***).
  + Для справки, указатели можно передавать в функцию по ссылке.
* Обратите внимание, что статическое поле инициализируется по-особенному в Visual Studio 2019, и обязано быть изначально инициализированным.
* При наличии компонентных данных типа указатель и их полной инициализации внутри класса, необходимо обеспечить корректную передачу объектов самого класса по значению в какую-либо функцию.
* Как мы знаем, передавая объект нашего класса в функцию по значению, в памяти создается его копия, и область видимости этой копии ограничена этой функцией. Как только функция будет выполнена, копия объекта нашего класса будет высвобождена из памяти.
* Конструктор по умолчанию, как и деструктор при компиляции будут создаваться автоматически, даже если их и не объявлять, вспоминаем первую лабораторную работу.
* Копия объекта класса, не имеющего в объявлении конструкторов, полностью копирует значения компонентных данных.
* Из этого всего следует, что требуется указать как правильно создавать копию объекта нашего класса, во избежание потери данных типа указатель. В противном случае, копия указателя будет инициализирована тем же адресом, что и у оригинального объекта. Уничтожая копию объекта, мы уничтожим так же и данные, на которые ссылаются указатели у оригинального объекта.

**Приложение 1**

**Варианты:**

1. Вычислительные машины
2. Строительные материалы
3. Строительные конструкции
4. Электроинструмент
5. Железнодорожный транспорт
6. Автомобильный транспорт
7. Воздушный транспорт
8. Водный транспорт
9. Энергетические установки
10. Запоминающие устройства
11. Устройства для управления компьютером
12. Телефонные аппараты
13. Бытовая техника
14. Устройства передачи данных
15. Программное обеспечение вычислительной техники
16. Печатающие устройства
17. Офисная мебель
18. Аппаратное обеспечение вычислительной техники
19. Элементная база для сборки компьютеров
20. Аппаратное обеспечение компьютерных сетей
21. Звуковоспроизводящая аппаратура
22. Звукозаписывающая аппаратура
23. Осветительные приборы
24. Оптические приборы
25. Нагревательные приборы
26. Холодильная техника
27. Электрические машины
28. Строительные машины
29. Металлургические машины
30. Сельскохозяйственные машины
31. Элементы интерьера
32. Канцелярские товары
33. Изделия целлюлозно-бумажной промышленности
34. Металлургические технологии
35. Врачебный контроль физиологического состояния спортсмена
36. Печи
37. Камины
38. Инструментальные среды для разработки программного обеспечения
39. Видеомониторы
40. Принтеры
41. Подъемно-транспортные машины
42. Плоттеры